软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2015,26(11):2994−3009 [doi: 10.13328/j.cnki.jos.004906]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 带间隔约束的 Top-*k* 对比序列模式挖掘*

杨 皓[1], 段 磊[1,3], 胡 斌[2], 邓 松[4], 王文韬[1], 秦 攀[1]

[1](四川大学 计算机学院,四川 成都 610065)

[2](国家电网 智能电网研究院,江苏 南京 210003)

[3](四川大学 华西公共卫生学院,四川 成都 610041)

[4](南京邮电大学 先进技术研究院,江苏 南京 210003)

通讯作者: 段磊, E-mail: leiduan@scu.edu.cn

**摘 要**: 对比序列模式能够表达序列数据集合间的差异,在商品推荐、用户行为分析和电力供应预测等领域有广泛的应用.已有的对比序列模式挖掘算法需要用户设定正例支持度阈值和负例支持度阈值.在不具备足够先验知识的情况下,用户难以设定恰当的支持度阈值,从而可能错失一些对比显著的模式.为此,提出了带间隔约束的 top-*k* 对比序列模式挖掘算法 *k*DSP-Miner(top-*k* distinguishing sequential patterns with gap constraint miner).*k*DSP-Miner 中用户只需设置期望发现的对比最显著的模式个数,从而避免了直接设置对比支持度阈值.相应地,挖掘算法更容易使用,并且结果更易于解释.同时,为了提高算法执行效率,设计了若干剪枝策略和启发策略.进一步设计了 *k*DSP-Miner 的多线程版本,以提高其对高维序列元素情况的处理能力.通过在真实世界数据集上的详实实验,验证了算法的有效性和执行效率.

**关键词**: 序列模式;top-*k*;对比挖掘

**中图法分类号**: TP311

中文引用格式: 杨皓,段磊,胡斌,邓松,王文韬,秦攀.带间隔约束的 Top-*k* 对比序列模式挖掘.软件学报,2015,26(11):2994−3009. http://www.jos.org.cn/1000-9825/4906.htm

英文引用格式: Yang H, Duan L, Hu B, Deng S, Wang WT, Qin P. Mining top-*k* distinguishing sequential patterns with gap constraint. Ruan Jian Xue Bao/Journal of Software, 2015,26(11):2994−3009 (in Chinese). http://www.jos.org.cn/1000-9825/4906.htm

## Mining Top-*k* Distinguishing Sequential Patterns with Gap Constraint

YANG Hao[1], DUAN Lei[1,3], HU Bin[2], DENG Song[4], WANG Wen-Tao[1], QIN Pan[1]

[1](School of Computer Science, Sichuan University, Chengdu 610065, China)

[2](Smart Grid Research Institute, State Grid, Nanjing 210003, China)

[3](West China School of Public Health, Sichuan University, Chengdu 610041, China)

[4](Institute of Advanced Technology, Nanjing University of Posts &Telecommunications, Nanjing 210003, China)

**Abstract**: Distinguishing sequential pattern can be used to present the difference between data sets, and thus has wide applications, such as commodity recommendation, user behavior analysis and power supply predication. Previous algorithms on mining distinguishing sequential patterns ask users to set both positive and negative support thresholds. Without sufficient prior knowledge of data sets, it is difficult for users to set the appropriate support thresholds, resulting in missing some significant contrast patterns. To deal with this problem, an algorithm, called *k*DSP-miner (top-*k* distinguishing sequential patterns with gap constraint miner), for mining top-*k* distinguishing sequential patterns satisfying the gap constraint is proposed. Instead of setting the contrast thresholds directly, a user-friendly parameter, which indicates the expected number of top distinguishing patterns to be discovered, is introduced in *k*DSP-miner.

* 基金项目: 国家自然科学基金(61103042); 中国博士后科学基金(2014M552371); 软件工程国家重点实验室开放研究基金(SKLSE2012-09-32)

收稿时间: 2015-05-31; 修改时间: 2015-07-14, 2015-08-11; 定稿时间: 2015-08-26

It makes *k*DSP-miner easy to use, and its mining result more comprehensible. In order to improve the efficiency of *k*DSP-miner, several pruning strategies and a heuristic strategy are designed. Furthermore, a multi-thread version of *k*DSP-miner is designed to enhance its applicability in dealing with the sequences with high dimensional set of elements. Experiments on real world data sets demonstrate that the proposed algorithm is effective and efficient.

**Key words**: sequential pattern; top-*k*; contrast mining

自 Agrawal 和 Srikant[1]提出序列模式挖掘以来,序列模式作为数据挖掘一项重要任务,吸引了大批研究者的关注,多种不同的序列模式陆续被提出来,如频繁序列模式[2]、对比序列模式[3]、闭合模式[4]、偏序模式[5]、周期模式[6]等等.在实际生活中,序列模式有着广泛的应用,例如,卫生疾控部门可以挖掘传染病传播在时间序列上的模式,其挖掘结果可用于发现传染病时空聚集性暴发规律,进而为防控工作提供参考;生物科学家可以通过分析 DNA 和蛋白质序列寻找疾病产生的根源,研发新药物;电力公司通过分析历史用电数据,提高对电力负荷预测的准确度.

序列挖掘中广泛使用的间隔约束的概念,让模式的匹配更加灵活.间隔约束是一个由两个非负整数确定的区间,表示序列模式中两个相邻元素间允许间隔的元素数目的最小值和最大值.例如,令间隔约束为[1,3],序列模式 $P=at$,若 $P$ 能够在序列 $S$ 中匹配,意味 $S$ 中存在元素 $a$ 和元素 $t$,并且存在一组 $at$,$a$ 在 $t$ 之前,且两者之间最少间隔 1 个元素,最多间隔 3 个元素.

现有的关于对比序列模式的挖掘工作,如文献[3,7],都需要用户预先设定正例支持度阈值$\alpha$、负例支持度阈值$\beta$和间隔约束$\gamma$.其目标是在间隔约束下,挖掘出正例支持度大于等于$\alpha$,并且负例支持度小于等于$\beta$的最小化模式.但是,这类挖掘算法有两个问题:(a) 用户很难设定合适的支持度阈值,如果设定了不合适的支持度阈值,挖掘出的模式可能不满足用户的期望;(b) 使用最小化约束进行剪枝,虽然减少了搜索空间,但可能会丢失一些对比显著的模式.我们以文献[3]提出的算法为例进行说明(文献[7]是基于文献[3]研究工作的扩展).

**Table 1** A gene sequence data set with two classes

**表 1** 含两个类别的基因序列数据集

| 序列 | 类别 | 序列 | 类别 |
|---|---|---|---|
| *a*, *g*, *c*, *t*, *g*, *a* | $D_+$ | | $D_-$ |
| *a*, *t*, *a*, *c*, *g*, *t* | | *g*, *a*, *g*, *a*, *t*, *c* | |
| *t*, *t*, *a*, *a*, *g*, *t* | | *t*, *g*, *c*, *a*, *g* | |
| *a*, *g*, *a*, *c*, *t*, *g* | | *g*, *g*, *g*, *c*, *a* | |
| *c*, *a*, *g*, *t*, *g* | | *a*, *g*, *c*, *g*, *a* | |
| *c*, *a*, *g*, *t*, *a* | | | |

例 1:表 1 示例了一个含有两个类别的基因序列数据集,其中,$D_+$代表正例数据集,$D_-$代表负例数据集.令$\gamma$=[0,2],运用文献[3]提出的算法挖掘对比模式,即,在 $D_+$中支持度大于等于$\alpha$,在 $D_-$中支持度小于等于$\beta$,且满足$\gamma$的最小化序列.

设置$\alpha$=0.9,$\beta$=0.1,挖掘结果为空集;将$\alpha$降低到 0.83,可以得到模式 *ct*;继续将$\alpha$降低至 0.66,或者将$\beta$增大为 0.25 才能挖掘到新的模式.由此可见,在不具备先验知识的情况下,用户并不容易设定合适的支持度阈值($\alpha$,$\beta$).

设置$\alpha$=1.0,$\beta$=0.25 时,可得到两个模式:*at* 和 *gt*,候选模式 *agt* 的正例支持度为 1.0,负例支持度为 0.25,因此满足支持度约束.但由于模式 *agt* 是模式 *at* 和模式 *gt* 的超序列,根据最小化模式约束,文献[3]算法会将其剪掉,于是,模式 *agt* 不会出现在结果集中.由此可见,最小化模式约束可能导致文献[3]算法丢失一些对比显著的模式.

从例 1 可以看出,不恰当的支持度阈值和最小化约束可能导致不能发现所有对比显著的模式.反复调整支持度阈值,必然增加用户使用算法的难度.故本文引入 top-*k* 约束,挖掘在全部对比模式中支持度对比最显著的 *k* 个对比序列模式.从分析对象和结果来看,挖掘 top-*k* 对比序列模式与挖掘满足支持度阈值的最小对比模式很相似,但是它们之间有本质的区别,具体表现在:

- 挖掘目标不同:带间隔约束的 top-*k* 对比序列模式是指支持度变化最为显著的 *k* 个对比序列模式,而带间隔约束的满足支持度阈值的对比模式指的是满足支持度阈值的对比序列模式.

- 挖掘过程不同:带间隔约束的 top-$k$ 对比序列模式不会使用最小化约束来进行剪枝,保证了支持度变化显著的对比序列模式不会由于其子序列满足条件而被剪掉.
- 应用难度不同:在挖掘带间隔约束的 top-$k$ 对比序列模式时,用户仅需设定期望得到的模式的数目,而不需要设定具体的正例支持度阈值和负例支持度阈值.一般来说,设定 $k$ 值比较直观,因而不会由于没有足够的先验知识,设定不合适的支持度阈值,导致丢失一些对比显著的模式.

据我们所知,目前还没有工作在带间隔约束的对比序列模式挖掘的基础上引入 top-$k$ 的概念.带间隔约束的 top-$k$ 对比序列模式挖掘旨在发现在两个数据集之间支持度变化最为显著的 $k$ 个对比序列模式,该方法可以避免由于不合适的阈值引起的对比显著的模式的错失;仅需用户设定期望得到的模式的数目即可,使用难度较以往的方法大大降低;同时,增强了挖掘结果的可解释性.本文主要工作包括:(a) 分析了设置正例支持度阈值和负例支持度阈值来挖掘对比序列模式的不足之处;(b) 提出了带间隔约束的 top-$k$ 对比序列模式挖掘问题;(c) 设计了带间隔约束的 top-$k$ 对比序列模式挖掘算法 $k$DSP-Miner(top-$k$ distinguishing sequential patterns with gap constraint miner);(d) 针对序列元素集合为高维的应用,设计了 $k$DSP-Miner 的多线程版本 MT-$k$DSP-Miner (multi-thread $k$DSP-miner);(e) 在真实数据集上,验证了提出算法的有效性和执行效率.

本文第 1 节介绍相关工作.第 2 节定义本文的研究问题.第 3 节讲述本文设计的带间隔约束的 top-$k$ 对比序列模式挖掘算法和其多线程版本的设计.第 4 节在真实世界的人类活动记录数据集、DNA 数据集和 Bible 数据集上验证本文提出算法的有效性和执行效率.第 5 节对本文的工作进行总结,并介绍下一步的工作方向.

## 1 相关工作

由于在众多领域的研究中发挥着重要作用,近年来,序列模式挖掘得到了越来越多的关注.文献[8−10]研究了蛋白质和核苷酸序列分析的问题;文献[11]将序列模式挖掘方法应用于软件缺陷的特征发现;文献[12]对音乐数据进行了深入分析;文献[13]提出了一个与语言无关的关键字抽取算法,并且不需要语义字典.

在序列模式中考虑间隔约束,可以提高序列模式的灵活性,扩大序列模式的应用范围.因此,挖掘带间隔约束的序列模式是序列模式挖掘的一个常见问题.文献[14]针对频繁闭合序列模式挖掘问题设计了 Gap-BIDE 的算法,该算法不需要维持候选集就可高效地得到所有带间隔约束的频繁闭合序列模式.文献[6]解决了在一条序列中挖掘带间隔约束的频繁周期序列模式的问题.文献[15]针对单条序列样本,研究了从中挖掘满足 one-off 条件和间隔约束的频繁模式的问题.

对比序列模式能够描述两类序列样本中的对比信息,识别不同类别样本集合的特征,适用于多个领域的序列模式分析.文献[3]设计了 ConSGapMiner 以发现最小化的对比序列模式.文献[16]建立了一个以对比模式作为特征的分类器,应用在多肽折叠预测问题上.文献[17]在对比序列模式挖掘中加入了“密度”的概念,并设计了算法挖掘满足“密度”约束的对比序列模式.文献[7]将基于单元素序列的最小化对比序列模式挖掘扩展为基于元素集合序列的最小化对比序列模式挖掘.

在数据挖掘应用中,经常可以得到大量的模式,top-$k$ 约束可以在所有的结果中选择最满足用户期望条件的 $k$ 个模式,简化挖掘的运行参数设置,提高结果的可解释性.文献[18]提出了在设定最小支持度的基础上挖掘 top-$k$ 频繁闭合序列模式.文献[19]挖掘两个数据集中的显露序列,使用距离而不是支持度来度量一个序列的显著程度.文献[20]在不确定的流数据上使用滑动窗口模型和 Chernoff 边界近似的对模式进行计数,以发现 top-$k$ 模式.

据我们所知,与本文最为相似的工作是文献[3],但文献[3]与本文有如下不同:首先,文献[3]旨在挖掘满足支持度阈值的最小化对比序列模式,而本文旨在挖掘支持度变化最为显著的 top-$k$ 对比序列模式;其次,文献[3]采用最小化模式约束,可能会导致无法发现一些对比显著的模式,而本文不会;再有,文献[3]需要设定正例样本的最小支持度阈值与负例样本的最大支持度阈值,而本文仅需设置期望发现的对比最显著的模式个数;最后,本文设计了多线程以提高算法的适用性,而文献[3]没有.因此,本文提出的算法相对于文献[3]而言,使用更加方便和直观,不需要对数据集有先验知识,也不需要多次调整支持度阈值来发现模式,因而适用性也更强.

## 2 问题定义

给定一个有穷元素集合$\Sigma$,将其称为字母表.由$\Sigma$中的元素构成的有序元素列称为序列,表示为 $S=e_1,e_2,\ldots,e_n$.对于任意 $i\in[1,n]$,有 $e_i\in\Sigma$成立.使用 $len(S)$来表示序列 $S$ 的长度,即,$S$ 中包含的元素的数目.为了方便表述,使用 $S[i]$来表示序列 $S$ 中第 $i$ 个位置上的元素($1\leqslant i\leqslant len(S)$).对于序列 $S$ 上的两个元素 $S[i]$和 $S[j]$($1\leqslant i<j\leqslant len(S)$),出现在 $S[i]$和 $S[j]$之间的元素个数称为 $S[i]$和 $S[j]$之间的间隔,记为 $Gap(S,i,j)=j-i-1$.

例 2:DNA 序列的字母表$\Sigma=\{a,c,g,t\}$,对于 DNA 序列 $S=a,t,a,c,g,t,len(S)=6,S[1]=a,S[5]=g,S[1]$与 $S[5]$之间的间隔 $Gap(S,1,5)=5-1-1=3$.

对于两个序列 $S$ 和 $S'(len(S)\geqslant len(S'))$,若存在一系列整数 $1\leqslant k_1<k_2<\ldots<k_{len(S')}\leqslant len(S)$,使得 $S'[i]=S[k_i]$对于 $i\in[1,len(S')]$恒成立,则称 $S'$是 $S$ 的子序列,$S$ 是 $S'$的超序列,将$\langle k_1,k_2,\ldots,k_{len(S')}\rangle$称为序列 $S'$在序列 $S$ 中的一个实例.

间隔约束通常被定义为一个由两个非负整数确定的区间,表示为$\gamma=[\gamma.\min,\gamma.\max](0\leqslant\gamma.\min\leqslant\gamma.\max)$.若$\langle k_1,k_2,\ldots,k_{len(S')}\rangle$是序列 $S'$在序列 $S$ 中的一个实例,并且对于 $1\leqslant i\leqslant len(S')-1,\gamma.\min\leqslant Gap(S,k_i,k_{i+1})\leqslant\gamma.\max$ 恒成立,则称 $S'$是 $S$ 在间隔约束$\gamma$下的子序列,表示为 $S'\subseteq_\gamma S$.为简洁起见,后文中的子序列和超序列都是指在间隔约束下的子序列和超序列.

例 3:对于序列 $S=a,g,c,t,g,a$ 和序列 $S'=a,g,\langle 1,2\rangle$和$\langle 1,5\rangle$都是 $S'$在 $S$ 中的实例.若令$\gamma=[0,2],0\leqslant Gap(S,1,2)=2-1-1=0\leqslant 2$,则 $S'\subseteq_\gamma S$.

给定序列集合 $D$,在间隔约束$\gamma$下,序列 $P$ 在数据集 $D$ 中的支持度由公式(1)得到:

$$\mathrm{Sup}(P,D,\gamma)=\frac{|\{S\in D\mid P\subseteq_\gamma S\}|}{|D|} \tag{1}$$

其中,$|D|$为集合 $D$ 中包含的序列的数目.

**定义 1(带间隔约束的对比度)**. 给定间隔约束$\gamma$、序列 $P$、正例数据集 $D_+$和负例数据集 $D_-$,$P$ 的对比度定义为

$$CR(P,D_+,D_-,\gamma)=\mathrm{Sup}(P,D_+,\gamma)-\mathrm{Sup}(P,D_-,\gamma) \tag{2}$$

后文中关于支持度和对比度的讨论都是在间隔约束$\gamma$下进行,简洁起见,将 $\mathrm{Sup}(P,D,\gamma)$简记为 $\mathrm{Sup}(P,D)$,将 $CR(P,D_+,D_-,\gamma)$简记为 $CR(P,D_+,D_-)$.

例 4:考虑表 1 中的 DNA 数据集,$|D_+|=6,|D_-|=4$.在间隔约束$\gamma=[0,2]$下,令 $P=agt$,那么 $\mathrm{Sup}(P,D_+)=1.0,\mathrm{Sup}(P,D_-)=0.25$,故 $CR(P,D_+,D_-)=1.0-0.25=0.75$.

给定间隔约束$\gamma$、正例数据集 $D_+$和负例数据集 $D_-$,若候选序列模式 $P$ 满足 $CR(P,D_+,D_-)>0$,则将 $P$ 称为一个带间隔约束的对比序列模式,简称为对比序列模式.

**定义 2(带间隔约束的 top-$k$ 对比序列模式)**. 给定间隔约束$\gamma$、正例数据集 $D_+$和负例数据集 $D_-$,在所有得到的对比序列模式中,对比度最大的 $k$ 个模式被称为带间隔约束的 top-$k$ 对比序列模式,简称为 top-$k$ 对比序列模式.

给定间隔约束$\gamma$、正例数据集 $D_+$和负例数据集 $D_-$,带间隔约束的 top-$k$ 对比序列模式挖掘的目标是:在满足$\gamma$的前提下,找出 $D_+$和 $D_-$之间对比度最大的 $k$ 个对比序列模式.

值得一提的是:在进行 top-$k$ 选择时,若有多个模式的对比度(公式(2)计算结果)相等,则按如下规则排序对比度:(1) 优先选择长度更短的模式;(2) 若长度相同,则按照算法中的元素顺序选择排序靠前的模式.在实践中,用户也可以根据实际情况修改规则,以适应具体应用的需求.

虽然本文与文献[3]都是关于对比序列模式挖掘的研究工作,但是本文的研究问题定义侧重于对比度的大小,并不要求对比序列模式满足最小化约束,因此能够挖掘出文献[3]中对比度显著(满足 $k$ 最大)但却不满足最小化约束的模式.

**命题 1**. 文献[3]的 ConSGapMiner 算法不适用于挖掘带间隔约束的 top-$k$ 对比序列模式.

证明:根据文献[3]的定义,给定模式 $P$ 和支持度阈值$\alpha,\beta$,ConSGapMiner 算法采用如下剪枝策略:

(1) 若 $\mathrm{Sup}(P,D_+)<\alpha$,那么 $P$ 及其超序列被剪掉;

(2) 若 $\mathrm{Sup}(P,D_+)\geqslant\alpha$,且 $\mathrm{Sup}(P,D_-)\leqslant\beta$,那么 $P$ 的超序列全部被剪掉.

令模式 $P'$满足 $len(P')=len(P)+1$ 且 $P'[i]=P[i](1\leqslant i\leqslant len(P))$,若有 $\mathrm{Sup}(P,D_+)=\mathrm{Sup}(P',D_+)$,$\mathrm{Sup}(P,D_-)>\mathrm{Sup}(P',D_-)$,那么 $CR(P,D_+,D_-)<CR(P',D_+,D_-)$.

由于 ConSGapMiner 算法采用深度优先遍历集合枚举树的方式产生候选,ConSGapMiner 算法会先产生模式 $P$,由于模式 $P$ 满足了支持度阈值条件,那么根据剪枝条件(2),模式 $P$ 的所有超序列将被剪掉.因此,$P'$不会出现在挖掘结果中.

所以,ConSGapMiner 算法不能保证发现所有带间隔约束的 top-$k$ 对比序列模式. □

例 5:考虑表 1 中的 DNA 数据集,令 $k$=5,间隔约束$\gamma$=[0,2],$D_+$为正例数据集,$D_-$为负例数据集,那么挖掘得到的带间隔约束的 top-$k$ 对比序列模式见表 2.

**Table 2** Top-5 distinguishing sequential patterns from gene sequence data set (Table 1) with gap constraint $\gamma$=[0,2]

**表 2** 基因序列数据集(表 1)的 top-5 对比序列模式(间隔约束$\gamma$=[0,2])

| 序号 | 模式 $P$ | $CR(P,D_+,D_-)$ | $\mathrm{Sup}(P,D_+)$ | $\mathrm{Sup}(P,D_-)$ |
|---|---|---|---|---|
| 1 | *ct* | 0.83 | 0.83 | 0.0 |
| 2 | *at* | 0.75 | 1.0 | 0.25 |
| 3 | *agt* | 0.75 | 1.0 | 0.25 |
| 4 | *gt* | 0.75 | 1.0 | 0.25 |
| 5 | *atg* | 0.67 | 0.67 | 0.0 |

为方便描述,表 3 中列出了本文常用的符号及定义.

**Table 3** Table of notations

**表 3** 符号表

| 符号 | 含义 |
|---|---|
| $\Sigma$ | 字母表,即一个有穷的元素集合 |
| $\gamma$ | 间隔约束 |
| $S'\sqsubseteq_\gamma S$ | $S'$是 $S$ 在间隔约束$\gamma$下的一个子序列 |
| $D_+$ | 正例数据集 |
| $D_-$ | 负例数据集 |
| $\mathrm{Sup}(P,D)$ | 在间隔约束$\gamma$下,模式 $P$ 在数据集 $D$ 中的支持度 |
| $CR(P,D_+,D_-)$ | 在间隔约束$\gamma$下,模式 $P$ 在数据集 $D_+$和 $D_-$之间的对比度 |

## 3 *k*DSP-Miner 算法设计

为了避免用户设定不合适的支持度阈值,导致不能发现对比显著的对比序列模式,本文提出在两个数据集之间挖掘支持度对比最显著的 $k$ 个对比序列模式的问题,并设计了一种带间隔约束的 top-$k$ 对比序列模式挖掘算法,将其命名为 *k*DSP-Miner.给定间隔约束,*k*DSP-Miner 可以挖掘出在正例数据集和负例数据集之间对比度最大的 $k$ 个对比序列模式.在给出 Naïve 算法的基础上,本文设计了 3 个剪枝策略以及 1 个启发策略,从而实现 *k*DSP-Miner.与 Naïve 算法相比,*k*DSP-Miner 算法的执行效率有明显的提升.针对序列元素集合高维的应用,本文在 *k*DSP-Miner 的基础上实现了 *k*DSP-Miner 的多线程版本,记为 MT-*k*DSP-Miner.接下来,我们首先给出 Naïve 算法;并以此为基础,介绍 *k*DSP-Miner 算法的关键技术和 MT-*k*DSP-Miner 的设计思路.

### 3.1 Naïve算法

Naïve 算法通过扫描数据集,生成字母表$\Sigma$;然后,利用$\Sigma$生成候选对比序列模式;最后输出结果.为了保证生成模式的完备性,即,找出带间隔约束的 top-$k$ 对比序列模式而不产生遗漏,Naïve 算法采用在序列数据模式挖掘中广泛使用的集合枚举树的方法[21]生成候选对比序列模式.给定字母表$\Sigma$,集合枚举树按照集合元素的全序,枚举集合中元素的所有可能组合.图 1 展示了对于基因数据集$\Sigma$={*a*,*c*,*g*,*t*},所生成的集合枚举树.

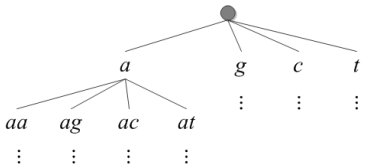



Fig.1 Demonstration of a set enumeration tree

图 1 集合枚举树示例

Naïve 算法以深度优先的方式遍历整个集合枚举树,在遍历过程中,逐一产生候选对比序列模式.

**性质 1**. 若候选模式 $P$ 满足 $\mathrm{Sup}(P,D_+)=0$,$P$ 及其超序列都不可能是 top-$k$ 对比序列模式.

证明:用反证法证明.

假设候选模式 $P$ 满足 $\mathrm{Sup}(P,D_+)=0$,且 $P$ 及其超序列是 top-$k$ 对比序列模式,即 $P\subseteq_\gamma P'$,使得:

$$CR(P',D_+,D_-)\geqslant \min\{CR(P'',D_+,D_-)|P''\in R\}>0,$$

其中,$R$ 为存储结果的集合.

因为候选模式 $P$ 满足 $\mathrm{Sup}(P,D_+)=0$,

所以,根据 Apriori 性质[1],对于所有的候选模式 $P\subseteq_\gamma P'$,$\mathrm{Sup}(P',D_+)\leqslant \mathrm{Sup}(P,D_+)=0$.

所以,$CR(P',D_+,D_-)=\mathrm{Sup}(P',D_+)-\mathrm{Sup}(P',D_-)\leqslant \mathrm{Sup}(P',D_+)-0=0$.

与假设矛盾.

所以,若候选模式 $P$ 满足 $\mathrm{Sup}(P,D_+)=0$,$P$ 及其超序列都不可能是 top-$k$ 对比序列模式. □

根据性质 1,可以得到遍历集合枚举树的停止条件:若候选模式 $P$ 满足 $\mathrm{Sup}(P,D_+)=0$,则剪去 $P$ 对应节点及其所有子孙节点.算法 1 给出了 Naïve 算法的伪代码.

**算法 1**. Naïve 算法.

输入:正例数据集 $D_+$、负例数据集 $D_-$、间隔约束 $\gamma$ 和参数 $k$.

输出:在间隔约束 $\gamma$ 下,$D_+$ 和 $D_-$ 之间对比度最大的 $k$ 个对比序列模式的集合 $R$.

```
1:  R←∅                                   //R 中仅保存对比度最大的 k 个对比序列模式
2:  Σ←scan(D+,D−)                         //扫描数据集 D+和 D−,生成数据集对应的字母表
3:  FOR 候选模式 P DO                      //按照深度优先遍历基于Σ的集合枚举树
4:    IF Sup(P,D+)=0 DO                   //遍历集合枚举树的停止条件
5:      剪去 P 的全部超序列;
6:    END IF
7:    IF CR(P,D+,D−)>0 DO
8:      IF |R|<k or |R|=k and CR(P,D+,D−)≥min{CR(P',D+,D−)|P'∈R} DO
9:        R←R∪{P};                        //如果|R|>k,从 R 中移除对比度最低的模式
10:     END IF
11:   END IF
12: END FOR
13: RETURN R
```

例 6:考虑的基因数据集(表 1),间隔约束 $\gamma=[0,2]$,候选模式 $P=att$,由于 $\mathrm{Sup}(P,D_+)=0$,所以 $P$ 的所有超序列,如 *atta*,*attc*,*attg*,*attt* 及它们的超序列都将被剪去.

### 3.2 剪枝和加速策略

Naïve 算法中,仅通过一个停止条件进行剪枝,遍历了大量无效的节点,这些节点对应的候选序列模式都不是 top-$k$ 对比序列模式.

**性质 2**. 如果一个元素 $e$ 只出现在负例数据集中,则所有包含元素 $e$ 的候选模式 $P$ 都不可能满足 top-$k$ 对比序列模式的定义.

证明:

因为元素 $e\in\Sigma$,$e\in D_-$,并且 $e\notin D_+$,

所以,$\mathrm{Sup}(e,D_+)=0$.

所以,根据 Apriori 性质,对于所有的候选模式 $e\subseteq_\gamma P$,$\mathrm{Sup}(P,D_+)\leqslant\mathrm{Sup}(e,D_+)=0$.

所以,$CR(P,D_+,D_-)=\mathrm{Sup}(P,D_+)-\mathrm{Sup}(P,D_-)\leqslant\mathrm{Sup}(P,D_+)-0\leqslant 0-0=0$.

所以,由定义 2 可知,top-$k$ 对比序列模式必须满足对比度大于 0.

所以,包含元素 $e$ 的候选模式 $P$ 不可能是 top-$k$ 对比序列模式. □

由性质 2 我们得到剪枝策略 1.

**剪枝策略 1(负例元素剪枝策略)**. 对于任意元素 $e\in\Sigma$,如果 $e\in D_-$,并且 $e\notin D_+$,将 $e$ 从 $\Sigma$ 中移除.

**性质 3**. 在 top-$k$ 对比序列模式挖掘过程中,当 $|R|=k$,令 $CR_k=\min\{CR(P'',D_+,D_-)|P''\in R\}$,且 $\mathrm{Sup}(P,D_+)<CR_k$,那么 $P$ 及其超序列都不是 top-$k$ 对比序列模式.

证明:假设结果集 $|R|=k$,令 $CR_k=\min\{CR(P'',D_+,D_-)|P''\in R\}$.

因为 $\mathrm{Sup}(P,D_+)<CR_k$,

所以,根据 Apriori 性质,对于所有的候选模式 $P\subseteq_\gamma P'$,$\mathrm{Sup}(P',D_+)\leqslant\mathrm{Sup}(P,D_+)<CR_k$.

所以,$CR(P',D_+,D_-)=\mathrm{Sup}(P',D_+)-\mathrm{Sup}(P',D_-)\leqslant\mathrm{Sup}(P',D_+)-0=\mathrm{Sup}(P',D_+)<CR_k$.

所以,候选模式 $P'$($P$ 及其超序列)的对比度均小于当前结果集中的最小对比度,必然小于最终结果集中的最小对比度,并且 $|R|=k$.

所以,$P$ 及其超序列不可能满足 top-$k$ 对比序列模式的定义. □

由性质 3 我们得到剪枝策略 2.

**剪枝策略 2(正例支持度 $k$ 剪枝策略)**. 在 top-$k$ 对比序列模式挖掘过程中,当 $|R|=k$,且在这 $k$ 个模式中,对比度的最小值为 $CR_k$ 时,若候选模式 $P$ 在正例数据集中的支持度小于 $CR_k$,即 $\mathrm{Sup}(P,D_+)<CR_k$,则剪去 $P$ 对应节点及其所有子孙节点.

**性质 4**. 在 top-$k$ 对比序列模式挖掘过程中,当 $|R|=k$ 时,令 $CR_k=\min\{CR(P',D_+,D_-)|P'\in R\}$,对于任意元素 $e\in\Sigma$,且 $\mathrm{Sup}(e,D_+)<CR_k$,所有包含元素 $e$ 的候选模式 $P$ 都不是 top-$k$ 对比序列模式.

证明:假设结果集 $|R|=k$,令 $CR_k=\min\{CR(P',D_+,D_-)|P'\in R\}$.

因为元素 $e\in\Sigma$,$\mathrm{Sup}(e,D_+)<CR_k$,

所以,根据 Apriori 性质,所有的包含元素 $e$ 的候选模式 $e\subseteq_\gamma P$,$\mathrm{Sup}(P,D_+)\leqslant\mathrm{Sup}(e,D_+)<CR_k$.

所以,$CR(P,D_+,D_-)=\mathrm{Sup}(P,D_+)-\mathrm{Sup}(P,D_-)\leqslant\mathrm{Sup}(P,D_+)-0=\mathrm{Sup}(P,D_+)<CR_k$.

所以,候选模式 $P'$(所有包含元素 $e$ 的候选模式)的对比度均小于当前结果集中的最小对比度,必然小于最终结果集中的最小对比度,并且 $|R|=k$.

所以,所有包含元素 $e$ 的候选模式不可能满足 top-$k$ 对比序列模式的定义. □

由性质 4 我们得到剪枝策略 3.

**剪枝策略 3(元素 $k$ 剪枝策略)**. 在 top-$k$ 对比序列模式挖掘过程中,当 $|R|=k$,且在这 $k$ 个模式中,对比度的最小值为 $CR_k$ 时,对于任意元素 $e\in\Sigma$,$\mathrm{Sup}(e,D_+)<CR_k$,剪去在集合枚举树中所有包含元素 $e$ 的节点.

在实践中我们发现,在上述 3 个剪枝策略中,负例元素剪枝策略(剪枝策略 1)可以在枚举过程之前使用;正例支持度 $k$ 剪枝策略(剪枝策略 2)一般来说是最有效的剪枝策略,但它和元素 $k$ 剪枝策略(剪枝策略 3)都是要找到 $k$ 个对比序列模式之后才开始生效,所以尽快找到对比度尽可能大的 $k$ 个对比序列模式能够加速整个枚举过程,减少不必要的尝试.

那么,怎样才能尽快找到对比度尽可能大的 $k$ 个对比序列模式呢?根据集合枚举树的性质,容易想到改变集合枚举树的枚举顺序,先枚举更可能产生对比序列模式的元素.为此,对 $\Sigma$ 中的元素排序:(a) 按 $CR(e,D_+,D_-)$ 从大

到小排序;(b) 按 Sup($e$,$D_+$)从大到小排序.依排序结果遍历集合枚举树,可望比在原来顺序下遍历集合枚举树更快地找到对比度大的对比序列模式,从而使剪枝策略 2 和剪枝策略 3 能够尽快生效,减少算法的运行时间.

**启发策略 1(排序加速策略)**. 对$\Sigma$中的所有元素按照 $CR(e,D_+,D_-)$降序,若相同,则按 Sup($e$,$D_+$)降序进行排列,排序结果记为$\Sigma'$.$k$DSP-Miner 算法按$\Sigma'$中元素的顺序遍历集合枚举树,生成候选对比序列模式.

实验结果表明,应用启发策略 1 能够有效地加速 $k$DSP-Miner 算法的执行效率(详见第 4.3 节).

值得一提的是:应用启发策略 1 并没有增加 $k$DSP-Miner 的计算量,因为$\Sigma$中的元素即为长度 1 的候选模式.此外,第 3.3 节将介绍:在多线程版本的 $k$DSP-Miner 中,$\Sigma$中元素的对比度也是线程计算任务分配的重要依据.

在上述剪枝策和启发策略基础上,算法 2 示例了 $k$DSP-Miner 算法的伪代码.

**算法 2**. $k$DSP-Miner 算法.

输入:正例数据集 $D_+$、负例数据集 $D_-$、间隔约束$\gamma$和参数 $k$.

输出:在间隔约束$\gamma$下,$D_+$和 $D_-$之间对比度最大的 $k$ 个对比序列模式的集合 $R$.

```
1:  R←∅                                  //R 中仅保存对比度最大的 k 个对比序列模式
2:  Σ←scan(D+)                           //仅仅扫描数据集 D+,生成候选元素集合,剪枝策略 1
3:  Σ'←sort(Σ)                           //对Σ进行排序,启发策略 1
4:  IF ∃e∈Σ and CR(e,D+,D-)>0
5:     R←R∪{e};                          //利用排序信息加快算法,如果|R|>k,移除 R 中对比度最小的模式
6:  END IF
7:  FOR 候选模式 P DO                    //按照深度优先遍历基于Σ'的集合枚举树(跳过长度为 1 的模式)
8:     IF Sup(P,D+)=0 DO                 //遍历集合枚举树的停止条件
9:        剪去 P 的全部超序列;
10:    END IF
11:    IF |R|=k and Sup(P,D+)<min{CR(P',D+,D-)|P'∈R} DO        //剪枝策略 2
12:       剪去 P 的全部超序列;
13:    END IF
14:    IF |R|=k and ∃e∈P and Sup(e,D+)<min{CR(P',D+,D-)|P'∈R} DO   //剪枝策略 3
15:       剪去所有包含 e 的节点;
16:    END IF
17:    IF CR(P,D+,D-)>0 DO
18:       IF |R|<k or |R|=k and CR(P,D+,D-)≥min{CR(P',D+,D-)|P'∈R} DO
19:          R←R∪{P};                   //如果|R|>k,从 R 中移除对比度最低的模式
20:       END IF
21:    END IF
22: END FOR
23: RETURN R
```

**定理 1**. 算法 2 能够找出对比度最大的 $k$ 个对比序列模式(完备性).

证明:使用反证法证明.

假设存在候选对比序列模式 $P$,满足 $CR(P,D_+,D_-)>\min\{CR(P'',D_+,D_-)|P''\in R\}$,且 $P\notin R$,$R$ 为最终的结果集.由于 $k$DSP-Miner 算法采用遍历集合枚举树的方式来生成对比序列模式,那么 $P$ 必然被剪枝策略 1、剪枝策略 2 或剪枝策略 3 剪掉.

如果是模式 $P$ 由剪枝策略 1 剪掉,那么存在 $e\in P$,且 $e\in D_-$,并且 $e\notin D_+$,所以,

$$\text{Sup}(P,D_+)=0, CR(P,D_+,D_-)\leqslant \text{Sup}(P,D_+)-0=0<\min\{CR(P'',D_+,D_-)|P''\in R\}.$$

与假设矛盾.

令 $R'$为 $P$ 被剪掉的时候对比度最大的 $k$ 个对比模式的集合,那么可得:

$$\min\{CR(P'',D_+,D_-)|P''\in R'\}\leqslant\min\{CR(P'',D_+,D_-)|P''\in R\}.$$

如果模式 $P$ 是由剪枝策略 2 剪掉,那么 $P'\subseteq_\gamma P$,且 $\mathrm{Sup}(P',D_+)<\min\{CR(P'',D_+,D_-)|P''\in R'\}$,所以,

$CR(P,D_+,D_-)\leqslant\mathrm{Sup}(P,D_+)-0=\mathrm{Sup}(P,D_+)\leqslant\mathrm{Sup}(P',D_+)<\min\{CR(P'',D_+,D_-)|P''\in R'\}\leqslant\min\{CR(P'',D_+,D_-)|P''\in R\}$.

与假设矛盾.

如果模式 $P$ 是由剪枝策略 3 剪掉,那么存在 $e\in P$,且 $e\in\Sigma$,并且 $\mathrm{Sup}(e,D_+)<\min\{CR(P',D_+,D_-)|P'\in R'\}$,所以,

$\mathrm{Sup}(P,D_+)\leqslant\mathrm{Sup}(e,D_+)$,

$CR(P,D_+,D_-)\leqslant\mathrm{Sup}(P,D_+)-0=\mathrm{Sup}(P,D_+)\leqslant\mathrm{Sup}(e,D_+)<\min\{CR(P',D_+,D_-)|P'\in R'\}\leqslant\min\{CR(P'',D_+,D_-)|P''\in R\}$.

与假设矛盾.

启发策略 1 仅改变集合枚举树的枚举顺序,其本身不会对集合枚举树进行剪枝,所以启发策略不会改变挖掘结果.

故算法 2 能够找出对比度最大的 $k$ 个对比序列模式. □

### 3.3 多线程*k*DSP-Miner算法设计

算法 2 主要由两部分构成:生成候选元素(步骤 2、步骤 3)和生成候选模式(步骤 4~步骤 22).其中,

- 生成候选元素包括对数据集进行扫描和对$\Sigma$进行排序,时间复杂度分别为 $O(|D_+|+|D_-|)$和 $O(|\Sigma|\log|\Sigma|)$,即,多项式级的;
- 生成候选模式需要遍历整个集合枚举树.理论上,给定最大长度 $l(l\geqslant 1)$,集合枚举树中包含$|\Sigma|^l$个候选模式.可见,对于具有高维$\Sigma$的序列集合,候选模式数量呈指数增长.

为了增强 $k$DSP-Miner 处理高维$\Sigma$序列数据的适用性,我们提出多线程版本的 $k$DSP-Miner,记为 MT-$k$DSP-Miner.设计思路的关键是:利用多线程技术实现候选模式对比度的并行计算.接下来,我们介绍 MT-$k$DSP-Miner 的设计细节.

典型情况是,对于含高维$\Sigma$的序列数据集合,$|\Sigma|>>m$($m$ 为线程总数).这样,以集合枚举树中 1 项集为根节点划分集合枚举树,可以得到$|\Sigma|$棵不相交的子树.MT-$k$DSP-Miner 以每棵子树为单位分配计算任务,因而不会重复计算,并且能保证算法的完备性.若出现$|\Sigma|<m$ 的情况,则需要进一步划分子树.

对$\Sigma$的所有元素(即 1 项集)按启发策略 1 进行排序得到$\Sigma'$,设元素 $e$ 在$\Sigma'$的排序为 $eid(1\leqslant eid\leqslant|\Sigma|)$,那么以 $e$ 为根节点的子树,被分配给编号为 $tid=(eid+1)\%m(1\leqslant tid\leqslant m)$的线程.这样,每个线程负责遍历大约$|\Sigma|/m$ 棵子树,每个线程被分配的任务大致相等,实现负载均衡.

例 7:对于表 1 的数据集,$\Sigma=\{a,c,g,t\}$,启发策略排序后为$\Sigma'=\{t,a,c,g\}$,若线程数目 $m$=2,那么 MT-$k$DSP-Miner 将集合枚举树以元素 $t$、元素 $c$ 分别为根节点的子树分配给一个线程,以元素 $g$、元素 $c$ 分别为根节点的子树分配给另一个线程.

不同于 $k$DSP-Miner 按深度优先顺序依次考察集合枚举树每个节点对应候选模式的对比度,MT-$k$DSP-Miner 对多棵子树并行遍历.需要注意的是,剪枝策略 2 和剪枝策略 3 依赖于当前全局的挖掘结果,为了让所有线程及时地运用剪枝策略 2 和剪枝策略 3,在 MT-$k$DSP-Miner 中,所有线程共享结果集.即,在挖掘过程中,所有线程同步更新当前 top-$k$ 的挖掘结果.

值得注意的是:线程数目并不是越多越好,因为 CPU 的计算能力有限,随着线程数目增加,MT-$k$DSP-Miner 算法运行时间总体上呈先下降,然后缓慢上升的趋势.实验中发现:在单机的环境下,当线程数目为 4~5 时,MT-$k$DSP-Miner 执行效率最高(详见第 4.4 节).

## 4 实 验

### 4.1 实验环境

本文在 4 组真实数据集上进行实验,以验证 $k$DSP-Miner 算法的有效性和执行效率以及 MT-$k$DSP-Miner 的

适用性.在实验采用的 4 组数据集中,Activity 和 Location 数据集是人类活动记录数据集,e.Coli 数据集为 DNA 数据集,这 3 个数据集都选自 UCI 机器学习数据集[22];而 Bible 数据集是文本型数据集.

Activity 和 Location 数据集分别是对文献[23]中的 ADLs 和 Sensor 数据集进行预处理后得到的数据集.将 ADLs 和 Sensor 数据集按照 End time 属性进行划分,每天的元组形成一条序列,然后抽取其中的 Activity 和 Location 属性形成的序列数据集.文献[23]数据包含 $A$ 和 $B$ 两个人的活动记录,这样就构成了 Activity+,Activity–和 Location+,Location–两组对比数据集.e.Coli 数据集包含有两个类型的 DNA 数据,可自然将其分成两类,构成数据集 e.Coli+和 e.Coli–.Bible 数据集包含代表旧约的 Bible+数据集和代表新约的 Bible–数据集,其中,圣经中的每个句子被当做一个序列,为了挖掘出有意义的模式,我们将诸如“the, a, is, was…”等常见停用词从数据集中去掉,最终得到 Bible 数据集.表 4 列出了实验中使用的数据集的特征.

**Table 4** Characteristic of data sets

**表 4** 数据集特征

| 数据集 $D$ | 类别 | 最小序列长度 | 最大序列长度 | 平均序列长度 | $\|\Sigma\|$ | $\|D\|$ |
|---|---|---|---|---|---|---|
| Activity | Activity+ | 13 | 26 | 17.71 | 9 | 14 |
| | Activity– | 16 | 43 | 23.48 | 10 | 21 |
| Location | Location+ | 2 | 43 | 27.27 | 12 | 15 |
| | Location– | 1 | 154 | 106.09 | 10 | 22 |
| e.Coli | e.Coli+ | 57 | 57 | 57 | 4 | 53 |
| | e.Coli– | 57 | 57 | 57 | 4 | 53 |
| Bible | Bible+ | 1 | 38 | 10.22 | 6 013 | 23 126 |
| | Bible– | 1 | 27 | 8.69 | 2 532 | 7 941 |

Naïve 算法、$k$DSP-Miner 算法和 MT-$k$DSP-Miner 算法均用 Java 实现,JDK 版本为 1.8.所有实验都在配置为 Intel Core i7-3770 3.90 GHz CPU,8GB 内存,Windows 7 操作系统的 PC 上完成.

### 4.2 有效性实验

为了验证 $k$DSP-Miner 算法的有效性,本文在 Activity,Location,e.Coli 和 Bible 数据集上分别运行 $k$DSP-Miner 算法,挖掘带间隔约束的 top-$k$ 对比序列模式.与本文最相似的工作是文献[3]提出的 ConSGapMiner 算法,给定的正例最小支持度阈值$\alpha$、负例最大支持度阈值$\beta$和间隔约束$\gamma$,ConSGapMiner 算法能够挖掘出在正例数据集中支持度大于等于$\alpha$、在负例数据集中支持度小于等于$\beta$,并满足间隔约束$\gamma$的最小化对比序列模式.

在 Activity,Location,e.Coli 和 Bible 这 4 个数据集上,以 Activity+/Activity–,Location+/Location–,e.Coli+/e.Coli–和 Bible+/Bible–作为正例数据集($D_+$)/负例数据集($D_-$).

$k$DSP-Miner 算法设定的参数为 $k$ 和$\gamma$,而 ConSGapMiner 算法需要设定参数$\alpha$,$\beta$和$\gamma$.为了公平地比较 $k$DSP-Miner 算法和 ConSGapMiner 算法,我们按如下方式设定两个算法的执行参数:令 $R_k$ 表示 $k$DSP-Miner 算法挖掘出的 top-$k$ 对比序列模式的集合,当 $k$DSP-Miner 算法参数设定为 $k$ 和$\gamma$,那么将 ConSGapMiner 算法的参数设定为正例支持度阈值$\alpha=\min\{\mathrm{Sup}(P,D_+)|P\in R_k\}$,负例支持度阈值$\beta=\max\{\mathrm{Sup}(P,D_-)|P\in R_k\}$,间隔约束为$\gamma$.这样,通过参照 $k$DSP-Miner 的挖掘结果设置 ConSGapMiner 算法的执行参数,从而使得两个算法的挖掘目标尽可能一致,从而使得结果具有可比性.

表 5 列出了实验中,$k$DSP-Miner 算法和 ConSGapMiner 算法的执行参数的对应关系.

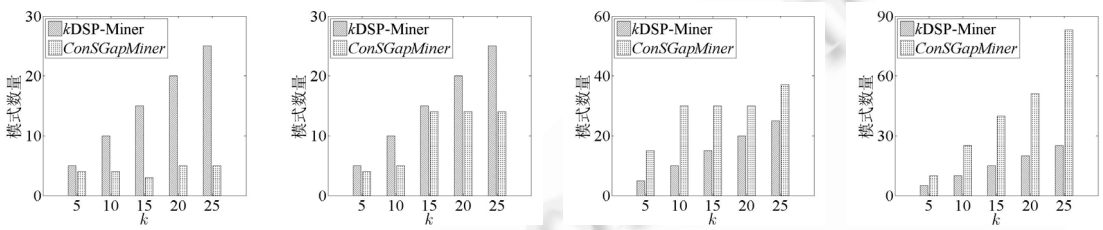
**Table 5** Support thresholds for ConSGapMiner w.r.t. values of $k$ in $k$DSP-Miner ($\gamma$=[0,2])

**表 5** $k$DSP-Miner 中,$k$ 值所对应的 ConSGapMiner 支持度阈值($\gamma$=[0,2])

| $k$ | Activity | | Location | | e.Coli | | Bible | |
|---|---|---|---|---|---|---|---|---|
| | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ | $\alpha$ | $\beta$ |
| 5 | 0.857 | 0.048 | 0.867 | 0.000 | 0.491 | 0.189 | 0.060 | 0.077 |
| 10 | 0.857 | 0.048 | 0.867 | 0.046 | 0.434 | 0.189 | 0.039 | 0.077 |
| 15 | 0.857 | 0.191 | 0.800 | 0.046 | 0.434 | 0.189 | 0.029 | 0.077 |
| 20 | 0.786 | 0.191 | 0.800 | 0.046 | 0.434 | 0.208 | 0.026 | 0.077 |
| 25 | 0.786 | 0.191 | 0.800 | 0.046 | 0.434 | 0.377 | 0.018 | 0.077 |

观察表 5 可以看出,在给定相同 $k$ 值的情况下,ConSGapMiner 算法中的支持度阈值 $\alpha$ 和 $\beta$ 在不同数据集上变化很大.可见,在不具备足够先验知识的情况下,对于 ConSGapMiner,用户难以设定合适的支持度阈值.

图 2 展示了 *k*DSP-Miner 算法和 ConSGapMiner 算法挖掘出的模式个数.*k*DSP-Miner 算法挖掘出的模式个数等于 $k$,可以发现,相应的 $\alpha$,$\beta$ 参数下,ConSGapMiner 算法挖掘出的模式个数不确定,可能大于 $k$ 个,也可能小于 $k$ 个.




(a) Activity (b) Location (c) e.Coli (d) Bible

Fig.2 Number of patterns of *k*DSP-Miner and ConSGapMiner ($\gamma$=[0,2])

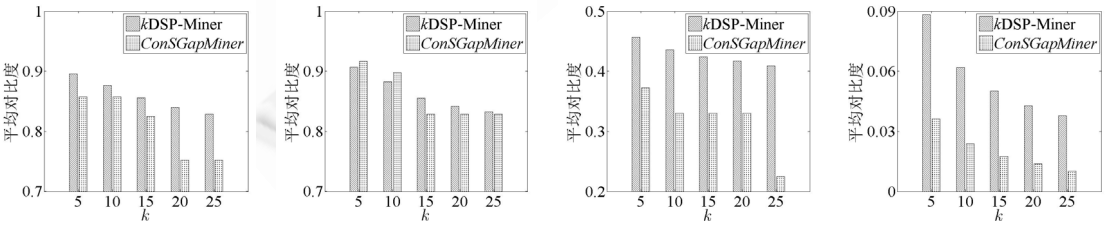
图 2 *k*DSP-Miner 和 ConSGapMiner 挖掘出的模式的个数($\gamma$=[0,2])

根据我们对 ConSGapMiner 参数的设置原则,若不存在最小化模式约束,则 ConSGapMiner 算法挖掘的结果应该是 *k*DSP-Miner 算法结果的超集.但由于 ConSGapMiner 算法具有最小化模式约束,导致某些满足支持度阈值而不满足最小化约束的模式被剪掉.但因其对比度满足 top-$k$ 的定义,故出现在 *k*DSP-Miner 算法的挖掘结果中而未出现在 ConSGapMiner 算法的结果集中,最终导致 *k*DSP-Miner 算法结果集中的模式多于 ConSGapMiner 算法的结果集中的模式.其中,在图 2(a)和图 2(b)中可以很明显见到这种情况.

对于图 2(d),我们可以发现,ConSGapMiner 算法发现模式的个数远大于 *k*DSP-Miner 算法发现模式的个数.这是由于 Bible 数据集中对比最显著的模式在正例和负例数据集中的支持度间隔很远,即,某一个模式,其对比显著,但其正例和负例支持度都远高(低)于其他模式的正例和负例的支持度.为了让 ConSGapMiner 算法尽可能地找全 *k*DSP-Miner 算法发现的模式,导致支持度阈值设置过于宽松,最终挖掘出大量的模式.

请注意,ConSGapMiner 算法由用户显式设置支持度阈值,无法预知能够找到多少个模式.若挖掘出的模式过少,则不利于揭示数据集蕴含的规律;若模式个数过多,则又会增加分析的难度.反复不断调整参数,必然增加用户使用算法的难度.相反,*k*DSP-Miner 算法通过设置期望发现模式的个数,避免了直接设定支持度阈值,非常直观,易于用户理解和使用.

由图 3 可知,在 4 个真实数据集的 20 组对比实验里,有 18 组实验中的 *k*DSP-Miner 算法的挖掘结果的平均对比度大于 ConSGapMiner 算法的执行结果.这是由于两种算法的目标模式并不相同:*k*DSP-Miner 算法挖掘对比度最为显著的 $k$ 个模式,而 ConSGapMiner 算法挖掘满足支持度阈值的模式.所以,即使将 ConSGapMiner 算法的支持度阈值设定为 *k*DSP-Miner 算法挖掘结果的支持度,其挖掘出的模式的平均对比度在大多数情况下也低于 *k*DSP-Miner 算法挖掘模式的平均对比度.




(a) Activity (b) Location (c) e.Coli (d) Bible

Fig.3 Average contrast ratio of patterns mined by *k*DSP-Miner and ConSGapMiner ($\gamma$=[0,2])

图 3 *k*DSP-Miner 和 ConSGapMiner 挖掘出的模式的平均对比度($\gamma$=[0,2])

有趣的是,我们发现,在图 3(b)的 $k$=5 和 $k$=10 的对比实验中,ConSGapMiner 算法挖掘模式的平均对比度大于 $k$DSP-Miner 算法挖掘模式的平均对比度.这是由于在 $k$=5 对应的参数下,由于 ConSGapMiner 算法要求的最小化模式约束,使得 ConSGapMiner 算法挖掘模式的个数少于 $k$DSP-Miner 算法挖掘结果,且其挖掘出的 4 个模式恰好具有最大对比度.在 $k$=10 对应的参数下,也发生了类似的情况.

为了进一步比较 $k$DSP-Miner 算法与 ConSGapMiner 算法的差异,我们用表 6 和表 7 分别列出在 $k$=10 这组对照参数下,$k$DSP-Miner 算法和 ConSGapMiner 算法在 Activity 数据集上的挖掘结果.比较表 6 中的模式 9 和表 7 中的模式 4 可以发现:由于 ConSGapMiner 算法有最小化约束,在找到满足支持度阈值的表 7 中的模式 4 后就不会继续搜索其超序列,因此,即使表 6 中的模式 9 具有很高的对比度,但是不会出现在 ConSGapMiner 算法的结果集中.比较表 6 中的模式 3 和表 7 中的模式 3 可以看出:表 6 中的模式 3 是表 7 中的模式 3 的超序列,虽然表 7 中的模式 3 的对比度比表 6 中的模式 3 的对比度大了 0.02,但是表 6 中的模式 3 的负例支持度为 0.发现负例支持度为 0 的对比序列模式,将有助于构建高性能的序列分类器.例如,文献[24]的研究表明,发现负例支持度为 0 的显露模式对提高分类精度非常有用.而 ConSGapMiner 算法并未挖掘出对比表 6 中的模式 3,即使表 5 中的模式 3 满足支持度阈值和间隔约束.

**Table 6** Top-10 distinguishing sequential patterns found by $k$DSP-Miner algorithm ($\gamma$=[0,2])

**表 6** $k$DSP-Miner 算法发现的带间隔约束的 top-10 对比序列模式($\gamma$=[0,2])

| 序号 | 模式 $P$ | $CR(P,D_+,D_-)$ | Sup$(P,D_+)$ | Sup$(P,D_-)$ |
|---|---|---|---|---|
| 1 | Sleeping, Toileting, Showering, Breakfast, Spare_Time/TV | 0.857 | 0.857 1 | 0.000 |
| 2 | Sleeping, Toileting, Showering, Breakfast | 0.857 | 0.857 1 | 0.000 |
| 3 | Sleeping, Showering Breakfast, Spare_Time/TV | 0.857 | 0.857 1 | 0.000 |
| 4 | Sleeping, Showering, Breakfast | 0.857 | 0.857 1 | 0.000 |
| 5 | Toileting, Showering, Breakfast, Spare_Time/TV | 0.857 | 0.857 1 | 0.000 |
| 6 | Toileting, Showering, Breakfast | 0.857 | 0.857 1 | 0.000 |
| 7 | Sleeping, Toileting, Showering, Spare_Time/TV | 0.881 | 0.929 | 0.048 |
| 8 | Sleeping, Showering, Spare_Time/TV | 0.881 | 0.929 | 0.048 |
| 9 | Showering, Breakfast, Spare_Time/TV | 0.929 | 0.929 | 0.000 |
| 10 | Showering, Breakfast | 0.929 | 0.929 | 0.000 |

**Table 7** Result of ConSGapMiner algorithm ($\alpha$=0.857, $\beta$=0.048, $\gamma$=[0,2])

**表 7** ConSGapMiner 算法挖掘结果($\alpha$=0.857,$\beta$=0.048,$\gamma$=[0,2])

| 序号 | 模式 $P$ | $CR(P,D_+,D_-)$ | Sup$(P,D_+)$ | Sup$(P,D_-)$ |
|---|---|---|---|---|
| 1 | Spare_Time/TV, Sleeping, Showering | 0.809 | 0.857 | 0.048 |
| 2 | Showering, Spare_Time/TV, Spare_Time/TV | 0.809 | 0.857 | 0.048 |
| 3 | Sleeping, Showering, Spare_Time/TV | 0.881 | 0.929 | 0.048 |
| 4 | Showering, Breakfast | 0.929 | 0.929 | 0.000 |

表 8 与表 9 分别展示了间隔约束$\gamma$和 $k$ 值对 $k$DSP-Miner 挖掘出来的对比序列模式的最小对比度(min $CR$)、最大对比度(max $CR$)和平均对比度(avg $CR$)的影响.从表 8 中可以看出:间隔约束$\gamma$变宽松,挖掘结果中的对比序列模式的最小对比度、最大对比度和平均对比度都呈现波动.这是由于间隔约束变宽,元素之间的组合变多,候选模式在正例和负例中的支持度都会增大,模式的对比度的变化趋势不能确定.

**Table 8** Contrast ratio of patterns under different gap constraints $\gamma$ ($k$=10)

**表 8** 不同间隔约束$\gamma$下对比度的值($k$=10)

| $\gamma$ | Activity | | | Location | | | e.Coli | | | Bible | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | min | max | avg | min | max | avg | min | max | avg | min | max | avg |
| [0,1] | 0.786 | 0.929 | 0.845 | 0.800 | 0.933 | 0.835 | 0.396 | 0.453 | 0.423 | 0.031 | 0.177 | 0.062 |
| [0,2] | 0.857 | 0.929 | 0.876 | 0.821 | 0.933 | 0.882 | 0.415 | 0.509 | 0.436 | 0.032 | 0.177 | 0.062 |
| [0,3] | 0.857 | 0.952 | 0.905 | 0.888 | 0.933 | 0.929 | 0.453 | 0.491 | 0.464 | 0.034 | 0.177 | 0.062 |
| [0,4] | 0.857 | 0.929 | 0.886 | 0.933 | 0.933 | 0.933 | 0.472 | 0.491 | 0.474 | 0.035 | 0.177 | 0.062 |
| [0,5] | 0.857 | 0.929 | 0.900 | 0.933 | 0.933 | 0.933 | 0.491 | 0.509 | 0.498 | 0.035 | 0.177 | 0.062 |

**Table 9** Contrast ratio of patterns under different $k$ ($\gamma$=[0,2])

**表 9** 不同 $k$ 值下对比度的值($\gamma$=[0,2])

| $k$ | Activity | | | Location | | | e.Coli | | | Bible | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | min | max | avg | min | max | avg | min | max | avg | min | max | avg |
| 5 | 0.857 | 0.929 | 0.895 | 0.867 | 0.933 | 0.907 | 0.434 | 0.509 | 0.457 | 0.044 | 0.177 | 0.088 |
| 10 | 0.857 | 0.929 | 0.876 | 0.821 | 0.933 | 0.882 | 0.415 | 0.509 | 0.436 | 0.032 | 0.177 | 0.062 |
| 15 | 0.810 | 0.929 | 0.856 | 0.800 | 0.933 | 0.855 | 0.396 | 0.509 | 0.424 | 0.022 | 0.177 | 0.050 |
| 20 | 0.786 | 0.929 | 0.839 | 0.800 | 0.933 | 0.841 | 0.396 | 0.509 | 0.417 | 0.020 | 0.177 | 0.043 |
| 25 | 0.786 | 0.929 | 0.829 | 0.800 | 0.933 | 0.833 | 0.377 | 0.509 | 0.409 | 0.017 | 0.177 | 0.038 |

令 $R_k$ 表示 top-$k$ 对比序列模式的集合,根据 top-$k$ 对比序列模式挖掘的定义可知:$R_k \subset R_{k+l}(l>0)$,且

$$\max\{CR(P,D_+,D_-)|P\in R_k\}=\max\{CR(P',D_+,D_-)|P'\in R_{k+l}\}.$$

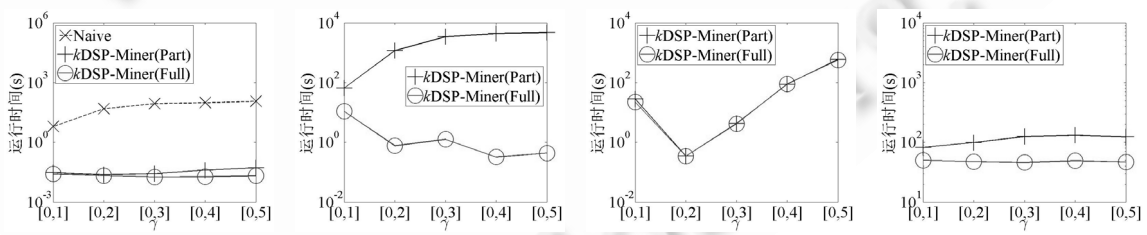
若 $P\notin R_k$,且 $P\in R_{k+l}(l>0)$,$CR(P,D_+,D_-)<\min\{CR(P',D_+,D_-)|P'\in R_k\}$,即,随着 $k$ 的增大,新加入结果集的对比序列模式的对比度呈非增长趋势,相应地,其平均对比度和最小对比度呈非增长趋势.表 9 的实验结果验证了此分析.

### 4.3 执行效率实验

为了验证 $k$DSP-Miner 的执行效率,本文使用 3 种算法进行对比:Naïve 算法、包含全部剪枝策略但不含启发策略的 $k$DSP-Miner(Part)以及包含全部剪枝和启发策略的 $k$DSP-Miner(Full).

图 4~图 6 展示了参数对算法执行效率的影响.为了更好地观察算法在不同参数下的执行效率,运行时间用对数形式表示.请注意:在 Location,e.Coli 和 Bible 数据集上,Naïve 算法的运行时间远大于 $k$DSP-Miner(Part)和 $k$DSP-Miner(Full)的运行时间,故未将其描绘在图上.

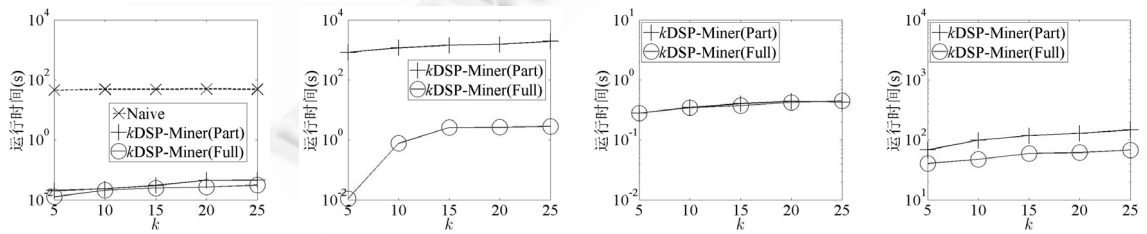
图 4 展示了当设置 $k$=10 时,间隔约束$\gamma$对算法执行效率的影响.随着间隔约束变宽,候选元素之间有效的组合变多,Naïve 算法运行时间会随之增加.对于 $k$DSP-Miner(Full)算法和 $k$DSP-Miner(Part)算法,候选模式在正例和负例的支持度都会随着间隔约束而变化,导致剪枝策略 2 和剪枝策略 3 生效的时间也开始变化.所以总体上,随着间隔约束变宽,$k$DSP-Miner(Full)算法和 $k$DSP-Miner(Part)算法的运行时间会变长;但是在某些间隔约束下,运行时间可能会降低.对于任意的间隔约束$\gamma$,$k$DSP-Miner(Full)算法快于 $k$DSP-Miner(Part)算法和 Naïve 算法.




(a) Activity (b) Location (c) e.Coli (d) Bible

Fig.4 Runtime under different gap constraints ($k$=10)

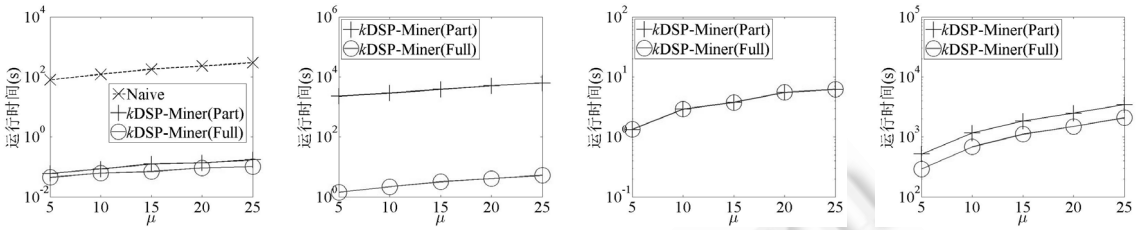
图 4 采用不同间隔约束的执行时间($k$=10)




(a) Activity (b) Location (c) e.Coli (d) Bible

Fig.5 Runtime under different values of $k$ ($\gamma$=[0,2])

图 5 采用不同 $k$ 值的执行时间($\gamma$=[0,2])




(a) Activity (b) Location (c) e.Coli (d) Bible

Fig.6 Scalability experiment ($k$=10,$\gamma$=[0,2])

图 6 伸缩性实验($k$=10,$\gamma$=[0,2])

图 5 展示了当设置$\gamma$=[0,2]时,$k$ 值对算法执行效率的影响.从图 5(a)可以看出:

- $k$ 值的变化对 Naïve 算法的运行时间没有影响,这是由于 Naïve 算法不含有与 $k$ 相关的剪枝策略;
- 对于 $k$DSP-Miner(Part)和 $k$DSP-Miner(Full)算法,由于剪枝策略 2 和剪枝策略 3 会在找到 $k$ 个候选对比序列模式后开始生效,所以 $k$ 越小,剪枝策略就越快开始工作,运行时间也就越少.所以,随着 $k$ 的增大,$k$DSP-Miner(Part)和 $k$DSP-Miner(Full)算法运行时间会增大.

为了测试数据集大小对算法执行效率的影响,本文将 Activity,Location,e.Coli 和 Bible 数据集进行复制,产生了大小为其$\mu$倍的数据集.图 6 展示了运行时间和参数$\mu$之间的关系,其中,$\gamma$=[0,2],$k$=10.可以看出:随着$\mu$的增加,3 种算法的运行时间都在增长,但 $k$DSP-Miner(Part)和 $k$DSP-Miner(Full)算法明显快于 Naïve 算法,且 $k$DSP-Miner (Full)算法快于 $k$DSP-Miner(Part)算法.

观察图 4(c)、图 5(c)和图 6(c)可以发现:在 e.Coli 数据集上,$k$DSP-Miner(Full)算法的效率和 $k$DSP-Miner(Part)算法很接近,表明启发策略 1 并没有很好地工作.这是因为 e.Coli 的字母表排序后的顺序和字典序是一致的,启发策略 1 对枚举顺序没有改变.而在另外 3 个数据集数据集上,启发策略 1 对算法执行效率都有明显的提升.特别地,观察图 4(b)、图 5(b)和图 6(b)可以发现,启发策略 1 在 Location 数据集上效果十分显著.

### 4.4 多线程算法实验

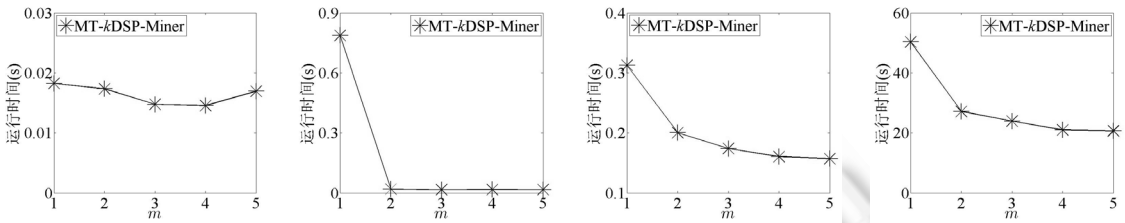
表 10 中给出了各个数据集上,在不同的 $k$ 值下,$k$DSP-Miner(Full)需要遍历的集合枚举树的节点数目($\gamma$=[0,2]).可以看出,随着 $k$ 值的增加,需要遍历的节点数目呈现增长趋势.所以,使用 MT-$k$DSP-Miner 将把对这些节点的遍历分配到多个线程上并行,是提高挖掘效率的有效途径.

**Table 10** Number of nodes traversed by $k$DSP-Miner in the set enumeration tree using different values of $k$

**表 10** 不同 $k$ 值下,$k$DSP-Miner 遍历节点的个数

| 数据集 | $k$ | | | | |
|---|---|---|---|---|---|
| | 5 | 10 | 15 | 20 | 25 |
| Activity | 1 395 | 6 147 | 12 393 | 18 261 | 27 999 |
| Location | 528 | 1 910 424 | 6 828 288 | 6 884 544 | 6 893 268 |
| e.Coli | 38 948 | 51 960 | 59 928 | 72 208 | 79 536 |
| Bible | 180 390 | 270 585 | 481 040 | 523 131 | 667 443 |

图 7 展示了在 4 个数据集上,在不同线程数目 $m$ 的情况下,MT-$k$DSP-Miner 的运行时间.其中,$k$=10,$\gamma$=[0,2].注意:当 $m$=1 时,MT-$k$DSP-Miner 的执行方式(单线程)等同于 $k$DSP-Miner 算法.图 7 表明:随着线程数目 $m$ 的增加,MT-$k$DSP-Miner 运行时间呈先减少再略微增加的趋势.这是由于随着线程数目的增加,每个线程所需要考察的候选模式的数量降低,相应地,执行时间减少.但当线程数目到一定程度后,受限于 CPU 的处理能力,线程多数时间处于等待 CPU 的状态,因而,增加线程数目不会降低执行时间,反而由于线程通信的开销,执行时间略微增加.




(a) Activity (b) Location (c) e.Coli (d) Bible

Fig.7 Runtime of MT-*k*DSP-Miner using different number of threads (*m*) (*k*=10,*γ*=[0,2])

图 7 采用不同线程数目(*m*),MT-*k*DSP-Miner 的运行时间(*k*=10,*γ*=[0,2])

## 5 结 论

对比序列模式常常被用于描述序列数据集的特征属性,但传统的对比序列模式挖掘需要预设支持度阈值,如果不能设定合适的阈值,挖掘出结果就可能不符合用户要求.针对这个问题,本文提出了带间隔约束的 top-*k* 对比序列模式挖掘问题,用户不需要直接预设具体的支持度阈值,只需确定期望挖掘出的最显著对比序列模式的个数.为了挖掘出带间隔约束的 top-*k* 对比序列模式,本文首先给出 Naïve 算法;然后,在 Naïve 算法的框架上设计了 3 个剪枝策略和 1 个启发策略,从而实现了高效的 *k*DSP-Miner 算法.为了进一步提高算法对序列元素集合为高维时的适用性,本文设计了 *k*DSP-Miner 的多线程版本 MT-*k*DSP-Miner.最后,在真实的人类活动记录数据集——DNA 数据集和 Bible 数据集上验证了 *k*DSP-Miner 算法的有效性和执行效率.

在下一步工作中,我们将进一步减少用户需要设定的参数,将间隔约束也设定为自动获取.本文中使用的数据集是元素的序列,还可以将其泛化为元素集合的序列,然后挖掘基于元素集的 top-*k* 对比序列模式.应用 *k*DSP-Miner 挖掘出来的结果来解决实际问题,如序列属性提取、构建分类器等.同时,针对大数据的时代背景,设计 *k*DSP-Miner 算法的 Hadoop 版本,进一步提高算法效率,然后将其应用于电力、交通和医学领域.

**References**:

[1] Agrawal R, Srikant R. Mining sequential patterns. In: Proc. of the 11th Int'l Conf. on Data Engineering. Washington: IEEE Computer Society Press, 1995. 3−14. [doi: 10.1109/ICDE.1995.380415]

[2] Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 2001,42(1-2):31−60. [doi: 10.1023/A:1007652502315]

[3] Ji X, Bailey J, Dong G. Mining minimal distinguishing subsequence patterns with gap constraints. Knowledge & Information Systems, 2007,11(3):259−286. [doi: 10.1007/s10115-006-0038-2]

[4] Yan X, Han J, Afshar R. CloSpan: Mining closed sequential patterns in large datasets. In: Proc. of the 3rd SIAM Int'l Conf. on Data Mining. SIAM, 2003. 166−177. [doi: 10.1137/1.9781611972733.15]

[5] Pei J, Wang H, Liu J, Wang K, Wang J, Yu PS. Discovering frequent closed partial orders from strings. IEEE Trans. on Knowledge & Data Engineering, 2006,18(11):1467−1481. [doi: 10.1109/TKDE.2006.172]

[6] Zhang M, Kao B, Cheung DW, Yip KY. Mining periodic patterns with gap requirement from sequences. ACM Trans. on Knowledge Discovery from Data, 2007,1(2):Article 7. [doi: 10.1145/1267066.1267068]

[7] Yang H, Duan L, Dong G, Nummenmaa J, Tang C, Li X. Mining itemset-based distinguishing sequential patterns with gap constraint. In: Proc. of the 21st Int'l Conf. of Database Systems for Advanced Applications. Switzerland: Springer-Verlag, 2015. 39−54. [doi: 10.1007/978-3-319-18120-2_3]

[8] Ferreira PG, Azevedo PJ. Protein sequence pattern mining with constraints. In: Proc. of the 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases. Berlin, Heidelberg: Springer-Verlag, 2005. 96−107. [doi: 10.1007/11564126_14]

[9] She R, Chen F, Wang K, Ester M, Gardy JL, Brinkman FSL. Frequent-Subsequence-Based prediction of outer membrane proteins. In: Proc. of the 9th ACM Knowledge Discovery and Data Mining. New York: ACM Press, 2003. 436−445. [doi: 10.1145/956750.956800]

[10] Wu X, Zhu X, He Y, Arslan AN. PMBC: Pattern mining from biological sequences with wildcard constraints. Computers in Biology & Medicine, 2013,43(5):481−492. [doi: 10.1016/j.compbiomed.2013.02.006]

[11] Zeng Q, Chen Y, Han G, Ren J. Sequential pattern mining with gap constraints for discovery of the software bug features. Journal of Computational Information Systems, 2014,10(2):673−680.

[12] Conklin D, Anagnostopoulou C. Comparative pattern analysis of Cretan folk songs. Journal of New Music Research, 2011,40(2): 119−125. [doi: 10.1080/09298215.2011.573562]

[13] Feng J, Xie F, Hu X, Li P, Cao J, Wu X. Keyword extraction based on sequential pattern mining. In: Proc. of the 3rd Int'l Conf. on Internet Multimedia Computing and Service. New York: ACM Press, 2011. 34−38. [doi: 10.1145/2043674.2043685]

[14] Li C, Yang Q, Wang J, Li M. Efficient mining of gap-constrained subsequences and its various applications. ACM Trans. on Knowledge Discovery from Data, 2012,6(1):74−77. [doi: 10.1145/2133360.2133362]

[15] Xie F, Wu X, Hu X, Gao J, Guo D, Fei Y, Hua E. MAIL: Mining sequential patterns with wildcards. Int'l Journal of Data Mining and Bioinformatics, 2013,8(1):1−23. [doi: 10.1504/IJDMB.2013.054690]

[16] Shah CC, Zhu X, Khoshgoftaar TM, Beyer J. Contrast pattern mining with gap constraints for peptide folding prediction. In: Proc. of the 21st Int'l Florida Artificial Intelligence Research Society Conf. Menlo Park: AAAI Press, 2008. 95−100.

[17] Wang X, Duan L, Dong G, Yu Z, Tang C. Efficient mining of density-aware distinguishing sequential patterns with gap constraints. In: Proc. of the 20th Int'l Conf. of Database Systems for Advanced Applications. Springer-Verlag, 2014. 372−387. [doi: 10.1007/978-3-319-05810-8_25]

[18] Han J, Wang J, Lu Y, Tzvetkov P. Mining top-$k$ frequent closed patterns without minimum support. In: Proc. of the 2002 Int'l Conf. on Data Mining. Washington: IEEE Computer Society Press, 2002. 211−218. [doi: 10.1109/ICDM.2002.1183905]

[19] Zaïane OR, Yacef K, Kay J. Finding top-$n$ emerging sequences to contrast sequence sets. Technical Report, TR07-03, Canada: Department of Computing Science, University of Alberta, 2007.

[20] Zhang X, Zhang Y. Sliding-Window top-$k$ pattern mining on uncertain streams. Journal of Computational Information Systems, 2011,7(3):984−992.

[21] Rymon R. Search through systematic set enumeration. In: Proc. of the 3rd Int'l Conf. on Principles of Knowledge Representation and Reasoning. San Francisco: Morgan Kaufmann Publishers, 1992. 539−550.

[22] Lichman M. UCI machine learning repository. 2013. http://archive.ics.uci.edu/ml

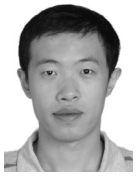
[23] Ordóñez FJ, Toledo P, Sanchis A. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. Sensors, 2013,13(5):5460−5477. [doi: 10.3390/s130505460]

[24] Dong G, Zhang X, Wong L, Li J. CAEP: Classification by aggregating emerging patterns. In: Proc. of the 2nd Int'l Conf. on Discovery Science. Berlin, Heidelberg: Springer-Verlag, 1999. 30−42. [doi: 10.1007/3-540-46846-3_4]

**杨皓**(1991－),男,四川资阳人,硕士生,CCF 学生会员,主要研究领域为数据挖掘.

**邓松**(1980－),男,博士,高级工程师,CCF 会员,主要研究领域为分布式数据挖掘,智能电网信息安全,电力信息物理融合系统.

**段磊**(1981－),男,博士,副教授,CCF 高级会员,主要研究领域为数据挖掘,进化计算,数据库技术.



**王文韬**(1991－),男,学士,主要研究领域为数据挖掘.

**胡斌**(1980－),男,高级工程师,主要研究领域为电力信息化,大数据及云计算技术在电力行业的应用研究.

**秦攀**(1991－),男,硕士生,主要研究领域为数据挖掘.